MACHINE LEARNING FOR TRADING

DAY-1 LAB EXPERIMENTS

1. **Mr Arun wants to start his own mobile phone company and he wants to wage an uphill battle with big smartphone brands like Samsung and Apple. But he doesn't know how to estimate the price of a mobile that can cover both marketing and manufacturing costs. So in this task, you don't have to predict the actual prices of the mobiles but you have to predict the price range of the mobiles. "**
    a. **Read the Mobile price dataset using the Pandas module**
    b. **print the 1st five rows.**
    c. **Basic statistical computations on the data set or distribution of data**
    d. **the columns and their data types**
    e. **Detects null values in the dataset. If there is any null values replaced it with mode value**
    f. **Explore the data set using heatmap**
    g. **Split the data in to test and train**
    h. **Fit in to the model Naive Bayes Classifier**
    i. **Predict the model**
    j. **Find the accuracy of the model**

CODE:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Step 1: Generate a Sample Dataset
# Create a DataFrame with simulated data
data = {
    'brand': np.random.choice(['BrandA', 'BrandB', 'BrandC'], size=100),
    'storage': np.random.choice([32, 64, 128, 256], size=100),
    'camera': np.random.choice(['Low', 'Medium', 'High'], size=100),
    'battery': np.random.choice([2000, 3000, 4000, 5000], size=100),
    'price_range': np.random.choice(['Low', 'Medium', 'High'], size=100)
}

df = pd.DataFrame(data)

# Step 2: Data Analysis
print("First five rows:")
print(df.head())

print("\nBasic statistical computations:")
print(df.describe(include='all'))

print("\nData types:")
print(df.dtypes)

# Detect null values
print("\nNull values in dataset:")
print(df.isnull().sum())
```

```
# Replace nulls with mode (though our dataset has none)
for column in df.columns:
    df[column].fillna(df[column].mode()[0], inplace=True)

# Step 3: Data Visualization using heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Heatmap of Features')
plt.show()

# Step 4: Encode categorical variables
df_encoded = pd.get_dummies(df, columns=['brand', 'camera'], drop_first=True)

# Step 5: Split data into training and testing sets
X = df_encoded.drop('price_range', axis=1)
y = df_encoded['price_range']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Fit Naive Bayes Classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Step 7: Make predictions
y_pred = model.predict(X_test)

# Step 8: Find the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy of the model: {accuracy:.2f}")
```

OUTPUT:

First five rows:

	brand	storage	camera	battery	price_range
0	BrandB	128	High	4000	Low
1	BrandC	64	High	5000	Low
2	BrandC	64	Medium	4000	High
3	BrandA	64	Low	5000	Medium
4	BrandC	256	Low	2000	High

Basic statistical computations:

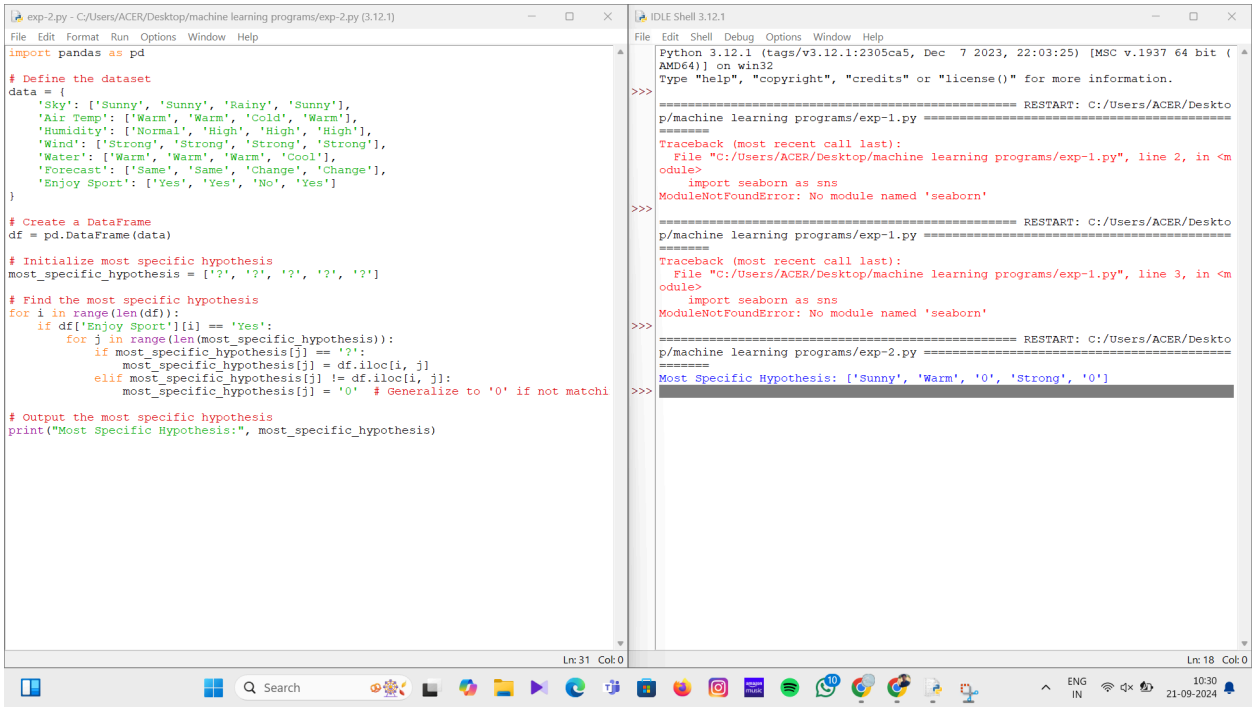
	brand	storage	camera	battery	price_range
count	100	100.000000	100	100.000000	100
unique	3	NaN	3	NaN	3
top	BrandC	NaN	High	NaN	Medium
freq	44	NaN	36	NaN	39
mean	NaN	102.400000	NaN	3520.000000	NaN
std	NaN	75.149951	NaN	1114.459601	NaN
min	NaN	32.000000	NaN	2000.000000	NaN
25%	NaN	32.000000	NaN	3000.000000	NaN
50%	NaN	64.000000	NaN	4000.000000	NaN
75%	NaN	128.000000	NaN	4250.000000	NaN
max	NaN	256.000000	NaN	5000.000000	NaN

Data types:
brand object
storage int64
camera object
battery int64
price_range object
dtype: object

Null values in dataset:
brand 0
storage 0
camera 0
battery 0
price_range 0
dtype: int64

2. **Implement a Python program for the most specific hypothesis using Find-S algorithm for the following given dataset and show the output:**

Example	Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



3. Develop a Python code for implementing Linear regression and show its performance

CODE:

```python
#experiment-3:
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Generate sample data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)  # Features
y = 4 + 3 * X + np.random.randn(100, 1)  # Target with some noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

OUTPUT:

```
Mean Squared Error: 0.92
R^2 Score: 0.65
```

4. Develop a Python code for implementing the KNN algorithm with an example.

CODE:

```python
#experiment-4:
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Create a simple dataset
data = {
    'Feature1': [1, 2, 1, 2, 3, 3, 4, 5],
    'Feature2': [2, 3, 4, 5, 1, 2, 3, 4],
    'Label': ['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B']
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Split data into features and labels
X = df[['Feature1', 'Feature2']]
y = df['Label']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Create and fit the KNN model
```

OUTPUT:

```
Accuracy: 1.00
```

5.provide short code: How is the Perception algorithm applied to the Iris flower classification problem? Anna is a botanist who is studying the Iris genus. She has collected data on the sepal length, sepal width, petal length, and petal width of various Iris flowers and wants to classify the flowers into their respective species based on their physical characteristics. Anna decides to use the Perception algorithm for this task.

CODE:

```
#experiment-5
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features: sepal length, sepal width, petal length, petal width
y = iris.target  # Labels: species

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and fit the Perceptron model
perceptron = Perceptron(max_iter=1000, tol=1e-3, random_state=42)
perceptron.fit(X_train, y_train)

# Make predictions
y_pred = perceptron.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
```

OUTPUT:

```
Accuracy: 0.47
```

6. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Example	Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

CODE AND OUTPUT:

```
import pandas as pd

# Define the dataset
data = {
    'Sky': ['Sunny', 'Sunny', 'Rainy', 'Sunny'],
    'Air Temp': ['Warm', 'Warm', 'Cold', 'Warm'],
    'Humidity': ['Normal', 'High', 'High', 'High'],
    'Wind': ['Strong', 'Strong', 'Strong', 'Strong'],
    'Water': ['Warm', 'Warm', 'Warm', 'Cool'],
    'Forecast': ['Same', 'Same', 'Change', 'Change'],
    'Enjoy Sport': ['Yes', 'Yes', 'No', 'Yes']
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Separate positive and negative examples
positive_examples = df[df['Enjoy Sport'] == 'Yes'].drop('Enjoy Sport', axis=1)
negative_examples = df[df['Enjoy Sport'] == 'No'].drop('Enjoy Sport', axis=1)

# Initialize most specific and most general hypotheses
specific_hypothesis = ['?', '?', '?', '?', '?']
general_hypothesis = ['0', '0', '0', '0', '0']

# Update specific hypothesis with positive examples
```

7. Develop a Python code for implementing Logistic regression and show its performance

CODE:

```python
#experiment-7:
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features: sepal length, sepal width, petal length, petal width
y = iris.target  # Labels: species

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and fit the Logistic Regression model
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, y_train)

# Make predictions
y_pred = log_reg.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

OUTPUT:

```
Accuracy: 1.00
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

8. Develop a Python code for implementing the Expectation Maximization algorithm with an example

CODE:

```
#experiment-8:
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

# Generate synthetic data
np.random.seed(0)
n_samples = 500

# Create two Gaussian distributions
X1 = np.random.normal(loc=0.0, scale=1.0, size=(n_samples, 2))
X2 = np.random.normal(loc=5.0, scale=1.0, size=(n_samples, 2))

# Combine the two datasets
X = np.vstack([X1, X2])

# Plot the generated data
plt.scatter(X[:, 0], X[:, 1], s=20)
plt.title('Generated Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()

# Apply the Expectation-Maximization algorithm using GMM
gmm = GaussianMixture(n_components=2, max_iter=100, random_state=0)
```

OUTPUT:


```
Cluster Means:
 [[ 5.02193878  5.00536543]
 [-0.06531522 -0.02507654]]
Cluster Covariances:
 [[[ 0.94716    -0.01781571]
  [-0.01781571  0.92727033]]

 [[ 0.94791051 -0.01002582]
  [-0.01002582  1.00011093]]]
```

DAY-2 LAB EXPERIMENTS

1. **Jack is a car enthusiast and wants to buy a new car. He wants to find the best deal and decides to use machine learning to predict the prices of different car models.Jack collects data on various features such as the make, model, year, engine size, and number of doors, as well as the sale price of each car. He splits the data into a training set and a test set and trains a linear regression model on the training data.Car Price Prediction with Machine Learning**
    a. **Read the dataset using the Pandas module**
    b. **print the 1st five rows.**
    c. **Basic statistical computations on the data set or distribution of data**
    d. **the columns and their data types**
    e. **Detects null values in the dataset. If there is any null values replaced it with mode value**
    f. **Explore the data set using heatmap**
    g. **Split the data in to test and train**
    h. **Fit in to the model Naive Bayes Classifier**
    i. **Predict the model**
    j. **Find the accuracy of the model**

CODE:

[9]
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Create a sample dataset
data = {
    'Make': ['Toyota', 'Honda', 'Ford', 'Chevrolet', 'Nissan'],
    'Model': ['Corolla', 'Civic', 'Mustang', 'Impala', 'Altima'],
    'Year': [2020, 2019, 2021, 2018, 2020],
    'Engine Size': [1.8, 2.0, 5.0, 3.6, 2.5],
    'Doors': [4, 4, 2, 4, 4],
    'Sale Price': [20000, 22000, 26000, 24000, 23000]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Print the first five rows
print(df.head())
```

```
[9]  # Explore dataset with heatmap
     plt.figure(figsize=(8, 5))
     sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
     plt.title('Correlation Heatmap')
     plt.show()

     # Prepare features and target variable
     X = df[['Year', 'Engine Size', 'Doors']]  # Features
     y = df['Sale Price']  # Target variable

     # Split into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3

     # Fit the model
     model = LinearRegression()
     model.fit(X_train, y_train)

     # Predict the model
     y_pred = model.predict(X_test)

     # Evaluate the model
     mse = mean_squared_error(y_test, y_pred)
     print(f"Mean Squared Error: {mse:.2f}")
```

OUTPUT:

```
        Make    Model  Year  Engine Size  Doors  Sale Price
0     Toyota  Corolla  2020          1.8      4       20000
1      Honda    Civic  2019          2.0      4       22000
2       Ford  Mustang  2021          5.0      2       26000
3  Chevrolet   Impala  2018          3.6      4       24000
4     Nissan   Altima  2020          2.5      4       23000
              Year  Engine Size     Doors    Sale Price
count     5.000000     5.000000  5.000000      5.000000
mean   2019.600000     2.980000  3.600000  23000.000000
std       1.140175     1.327403  0.894427   2236.067977
min    2018.000000     1.800000  2.000000  20000.000000
25%    2019.000000     2.000000  4.000000  22000.000000
50%    2020.000000     2.500000  4.000000  23000.000000
75%    2020.000000     3.600000  4.000000  24000.000000
max    2021.000000     5.000000  4.000000  26000.000000
Make            object
Model           object
Year             int64
Engine Size    float64
Doors            int64
Sale Price       int64
dtype: object
```

2. **Implement a Python program for the most specific hypothesis using Find-S:**

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

CODE:

```python
# Define the dataset
data = {
    'Origin': ['Japan', 'Japan', 'Japan', 'USA', 'Japan'],
    'Manufacturer': ['Honda', 'Toyota', 'Toyota', 'Chrysler', 'Honda'],
    'Color': ['Blue', 'Green', 'Blue', 'Red', 'White'],
    'Decade': ['1980', '1970', '1990', '1980', '1980'],
    'Type': ['Economy', 'Sports', 'Economy', 'Economy', 'Economy'],
    'Example Type': ['Positive', 'Negative', 'Positive', 'Negative', 'Positive']
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Initialize most specific hypothesis
specific_hypothesis = ['?' for _ in range(len(df.columns) - 1)]

# Iterate through the dataset
for index, row in df.iterrows():
    if row['Example Type'] == 'Positive':
        for i in range(len(specific_hypothesis)):
            if specific_hypothesis[i] == '?':
                specific_hypothesis[i] = row[i]
            elif specific_hypothesis[i] != row[i]:
                specific_hypothesis[i] = '0'  # Use '0' to indicate no match
```

OUTPUT:

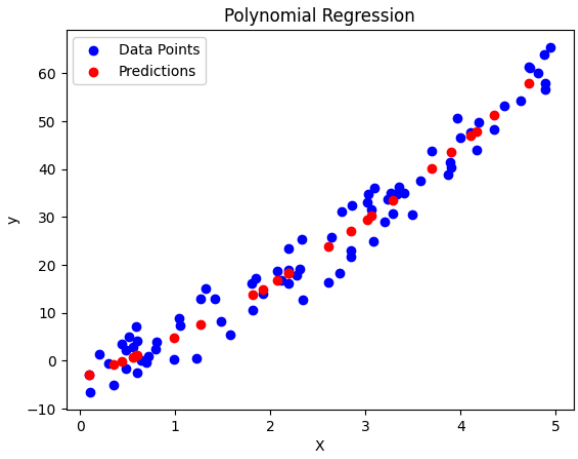
```
Most Specific Hypothesis: ['Japan', '0', '0', '0', 'Economy']
```

3. **Develop a Python code for implementing Polynomial regression and show its performance.**

CODE:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Generate synthetic data
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = 2 * X**2 + 3 * X + np.random.randn(80, 1) * 4

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Transform features to polynomial features
poly = PolynomialFeatures(degree=2)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

# Fit the model
model = LinearRegression()
model.fit(X_poly_train, y_train)
```

OUTPUT:



4. **Develop a Python code for implementing the KNN algorithm with an example**

   CODE:

```python
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Labels

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and fit the KNN model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT:

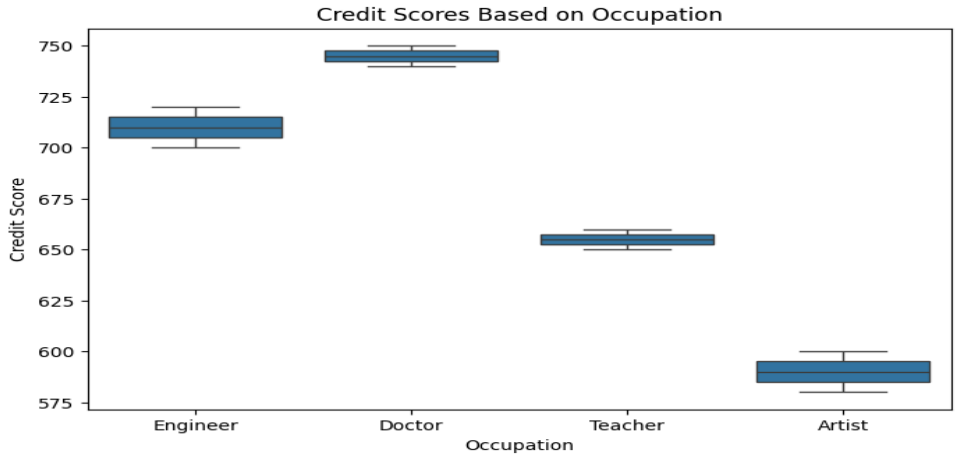
```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

5. **John is a young professional who wants to buy his first home. He knows that his credit score is an important factor in determining whether he will be approved for a loan, so he decides to check it. He goes to a financial website that offers a free credit score prediction service based on machine learning algorithms.Print the 1st five rows (b.) Basic statistical computations on the data set or distribution of data (c) The columns and their data types (d) Detects null values in the dataset. If there is any null values replaced it with mode value (e) Explore the data set using ps.box(Credit Scores Based on Occupation) (f) Split the data in to test and train (g) Fit in to the model Naive Bayes Classifier (i) Predict the model**

CODE:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

# Sample dataset creation
data = {
    'Occupation': ['Engineer', 'Doctor', 'Teacher', 'Artist', 'Engineer', 'Doctor', 'Teacher', 'Artist'],
    'Credit Score': [700, 750, 650, 600, 720, 740, 660, 580],
    'Age': [25, 30, 22, 28, 26, 31, 23, 27],
    'Loan Approved': ['Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No']
}

# Convert to DataFrame
df = pd.DataFrame(data)

# (a) Print the first five rows
print(df.head())

# (b) Basic statistics and distribution
print(df.describe())

# (c) Columns and their data types
```

OUTPUT:



6. **For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.**

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

CODE:

```python
# Define the dataset
data = {
    'Origin': ['Japan', 'Japan', 'Japan', 'USA', 'Japan'],
    'Manufacturer': ['Honda', 'Toyota', 'Toyota', 'Chrysler', 'Honda'],
    'Color': ['Blue', 'Green', 'Blue', 'Red', 'White'],
    'Decade': [1980, 1970, 1990, 1980, 1980],
    'Type': ['Economy', 'Sports', 'Economy', 'Economy', 'Economy'],
    'Example Type': ['Positive', 'Negative', 'Positive', 'Negative', 'Positive']
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Separate positive and negative examples
positive_examples = df[df['Example Type'] == 'Positive'].drop('Example Type', axis=1)
negative_examples = df[df['Example Type'] == 'Negative'].drop('Example Type', axis=1)

# Initialize most specific and most general hypotheses
specific_hypothesis = ['?', '?', '?', '?', '?']
general_hypothesis = ['0', '0', '0', '0', '0']

# Update specific hypothesis with positive examples
for index, row in positive_examples.iterrows():
    for i in range(len(specific_hypothesis)):
        if specific_hypothesis[i] == '?':
```

OUTPUT:

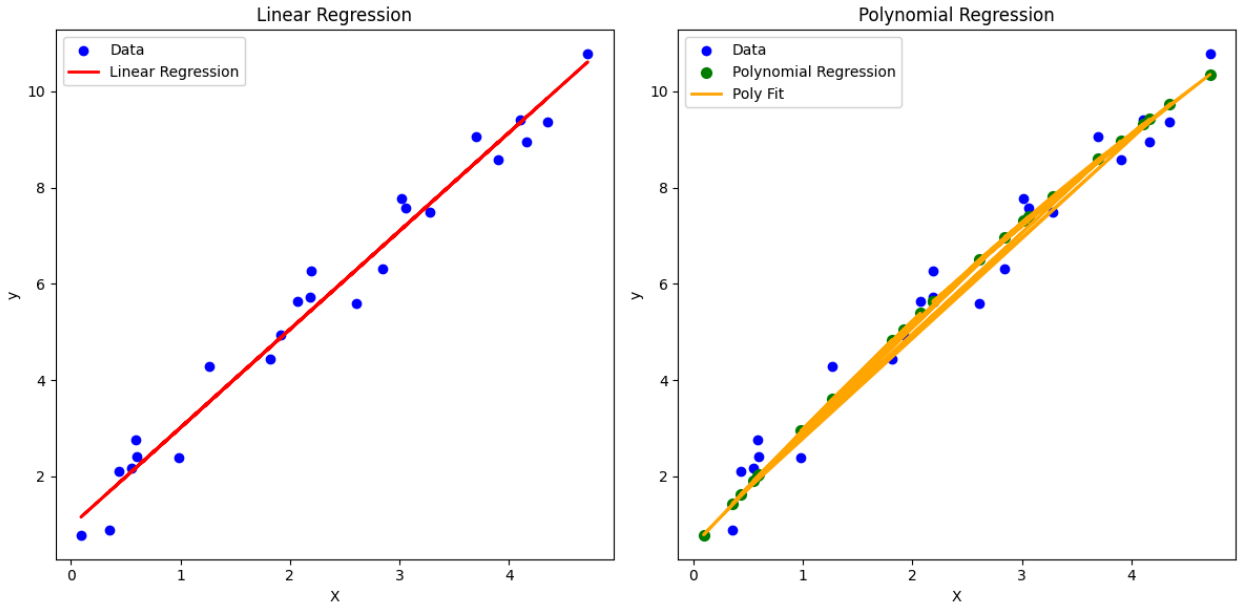
```
Most Specific Hypothesis: ['Japan', '0', '0', '0', 'Economy']
Most General Hypothesis: ['?', '?', '?', '?', '?']
```

7. **Develop a Python code for implementing Linear and Polynomial regression and show its performance**

CODE:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Generate synthetic data
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = 2 * X.flatten() + 1 + np.random.randn(80) * 0.5  # Linear relationship with noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)

# Polynomial Regression
poly_features = PolynomialFeatures(degree=2)
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)
```

OUTPUT:

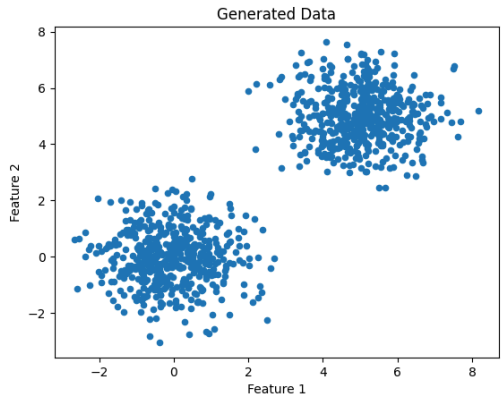


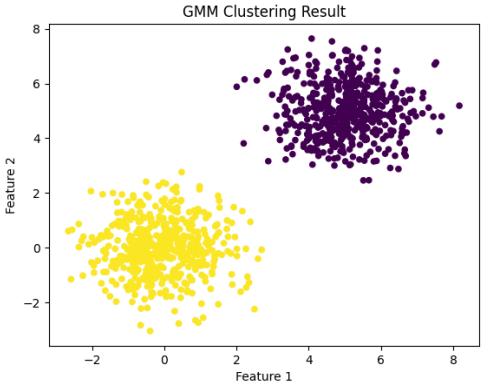
8. **Develop a Python code for implementing the Expectation Maximization algorithm with and example**

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

# Generate synthetic data
np.random.seed(0)
n_samples = 500

# Create two Gaussian distributions
X1 = np.random.normal(loc=0.0, scale=1.0, size=(n_samples, 2))
X2 = np.random.normal(loc=5.0, scale=1.0, size=(n_samples, 2))

# Combine the two datasets
X = np.vstack([X1, X2])

# Plot the generated data
plt.scatter(X[:, 0], X[:, 1], s=20)
plt.title('Generated Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()

# Apply the Expectation-Maximization algorithm using GMM
gmm = GaussianMixture(n_components=2, max_iter=100, random_state=0)
gmm.fit(X)
```

OUTPUT:





DAY-3 LAB EXPERIMENTS

1. **Sarah is a botanist who studies different species of plants. She is particularly interested in the Iris genus and has collected data on the sepal length, sepal width, petal length, and petal width of various Iris flowers. She wants to use this data to classify the flowers into their respective species based on their physical characteristics. Sarah decides to use a machine learning algorithm for this task and trains a model on her collected data. The algorithm uses the sepal and petal measurements as input features and predicts the species of the flower based on these features. One day, Sarah is out in the field collecting new samples of Iris flowers. She measures the sepal and petal characteristics of each flower and inputs this information into the trained model. The model then predicts the species of each flower based on its physical characteristics.**
    a. **Read the IRIS.csv Data set using the Pandas module**
    b. **Plot the data using a scatter plot "sepal_width" versus "sepal_length" and color species.**
    c. **Split the data**
    d. **Fit the data to the model**
    e. **Predict the model with new test data [5, 3 , 1, .3]**

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder

# Load the Iris dataset
df = pd.read_csv('IRIS.csv')

# Plot the data using a scatter plot: "sepal_width" vs "sepal_length" and color by species
species_encoder = LabelEncoder()
df['Species_encoded'] = species_encoder.fit_transform(df['Species'])

plt.figure(figsize=(8, 6))
plt.scatter(df['sepal_length'], df['sepal_width'], c=df['Species_encoded'], cmap='viridis')
plt.title('Sepal Width vs Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.colorbar(label='Species')
plt.show()

# Prepare features (X) and target (y)
X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df['Species']
```

OUTPUT:

2. **Implement a Python program for the most specific hypothesis using Find-S algorithm for the following given dataset and show the output:**

Example	Shape	Size	Color	Surface	Thickness	Target Concept
1	Circular	Large	Light	Smooth	Thick	Malignant (+)
2	Circular	Large	Light	Irregular	Thick	Malignant (+)
3	Oval	Large	Dark	Smooth	Thin	Benign (-)
4	Oval	Large	Light	Irregular	Thick	Malignant (+)

CODE:

```
data = [
    ['Circular', 'Large', 'Light', 'Smooth', 'Thick', 'Malignant'],
    ['Circular', 'Large', 'Light', 'Irregular', 'Thick', 'Malignant'],
    ['Oval', 'Large', 'Dark', 'Smooth', 'Thin', 'Benign'],
    ['Oval', 'Large', 'Light', 'Irregular', 'Thick', 'Malignant']
]

# Extract features and target concept
features = [example[:-1] for example in data]  # All attributes except the last one
target = [example[-1] for example in data]     # Last column (Target Concept)

# Initialize the most specific hypothesis
hypothesis = ['?' for _ in range(len(features[0]))]

# Find-S algorithm
for i in range(len(features)):
    if target[i] == 'Malignant':  # Only consider positive examples
        for j in range(len(hypothesis)):
            if hypothesis[j] == '?':
                hypothesis[j] = features[i][j]
            elif hypothesis[j] != features[i][j]:
                hypothesis[j] = '?'

# Output the final hypothesis
print("The most specific hypothesis is:", hypothesis)
```

OUTPUT:

```
The most specific hypothesis is: ['?', 'Large', 'Light', 'Irregular', 'Thick']
```

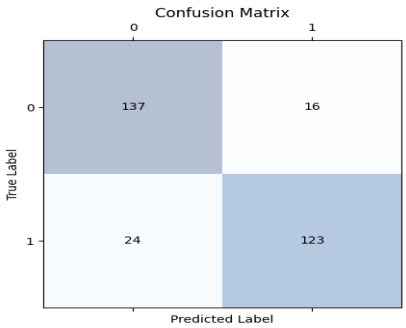
3. **Develop a Python code for implementing Logistic regression and show its performance**

CODE:

:

```
#experiment-3:
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification

# Generate a synthetic dataset for binary classification
X, y = make_classification(n_samples=1000, n_features=4, n_classes=2, random_state=42)

# Convert the dataset to a DataFrame
df = pd.DataFrame(X, columns=['Feature1', 'Feature2', 'Feature3', 'Feature4'])
df['Target'] = y

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df[['Feature1', 'Feature2', 'Feature3', 'Feature4']], df['Target'], test_size=0.3, random_state=42)

# Initialize the Logistic Regression model
logistic_model = LogisticRegression()

# Train the model
logistic_model.fit(X_train, y_train)
```

OUTPUT:



4. **Develop a Python code for implementing the Naive Bayes algorithm with and example**

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Convert to DataFrame for better understanding
df = pd.DataFrame(X, columns=iris.feature_names)
df['Species'] = y

# Print first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the Naive Bayes model
nb_model = GaussianNB()
```

OUTPUT:

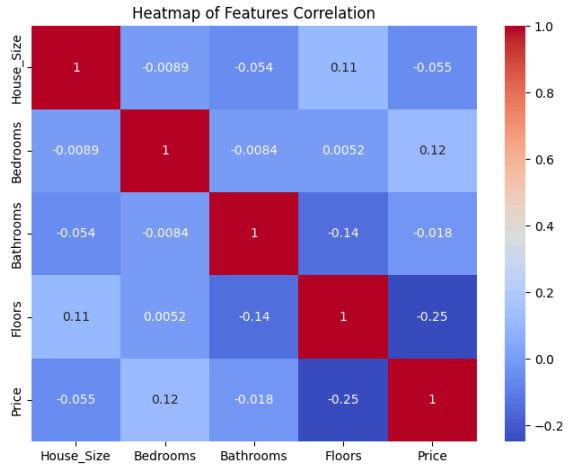
```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   Species
0        0
1        0
2        0
3        0
4        0

Accuracy: 97.78%

Confusion Matrix:
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      0.92      0.96        13
```

5. **Mark and his family are planning to move to a new city and are in the market for a new home. They have been searching online for homes in their desired area and have found several properties that meet their requirements. However, they are not sure about the prices of these homes and want to get a rough estimate before making an offer.How will you help Mark to buy a new house.**
   a. **Read the house Data set using the Pandas module**
   b. **Print the 1st five rows. Basic statistical computations on the data set or distribution of data .**
   c. **Print the columns and their data types.**
   d. **Detects null values in the dataset. If there is any null values replaced it with mode value.**
   e. **Explore the data set using heatmap .**
   f. **Split the data in to test and train .**
   g. **Predict the price of a house**

CODE:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Generate a synthetic dataset
np.random.seed(42)

# Create a DataFrame with synthetic data
data = {
    'House_Size': np.random.randint(1000, 4000, 100),
    'Bedrooms': np.random.randint(2, 6, 100),
    'Bathrooms': np.random.randint(1, 4, 100),
    'Floors': np.random.randint(1, 3, 100),
    'Price': np.random.randint(150000, 500000, 100)  # Target variable (house price)
}

df = pd.DataFrame(data)

# Print the first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())
```

OUTPUT:



6. **For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.**
   CODE:

```
data = [
    ['Circular', 'Large', 'Light', 'Smooth', 'Thick', 'Malignant'],
    ['Circular', 'Large', 'Light', 'Irregular', 'Thick', 'Malignant'],
    ['Oval', 'Large', 'Dark', 'Smooth', 'Thin', 'Benign'],
    ['Oval', 'Large', 'Light', 'Irregular', 'Thick', 'Malignant']
]

# Separate attributes and target concept
attributes = np.array([x[:-1] for x in data])  # Attributes (Shape, Size, etc.)
targets = np.array([x[-1] for x in data])      # Target concept (Malignant or Benign)

# Initialize the most specific hypothesis
specific_hypothesis = ['0', '0', '0', '0', '0']  # More specific hypothesis (initially most restrictive)

# Candidate Elimination: Find the most specific hypothesis for positive examples
for i, instance in enumerate(attributes):
    if targets[i] == 'Malignant':  # We only consider positive examples (Malignant)
        for j in range(len(specific_hypothesis)):
            if specific_hypothesis[j] == '0':  # First time, initialize with first positive instance
                specific_hypothesis[j] = instance[j]
            elif specific_hypothesis[j] != instance[j]:  # If different, generalize with '?'
                specific_hypothesis[j] = '?'

# Output the most specific hypothesis
print("Most Specific Hypothesis:", specific_hypothesis)
```

OUTPUT:

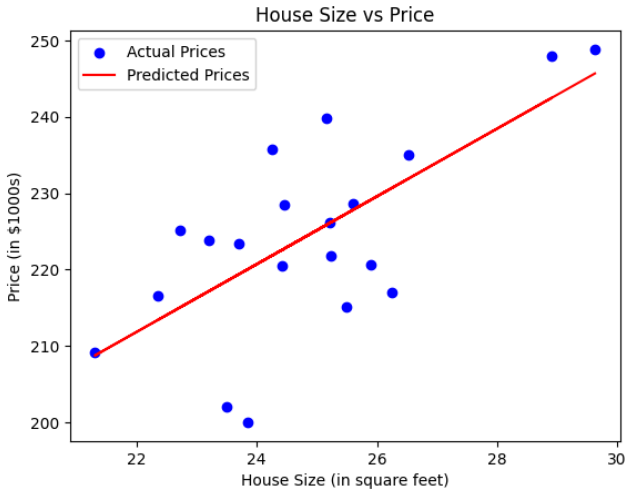
```
Most Specific Hypothesis: ['?', 'Large', 'Light', '?', 'Thick']
```

7. **Develop a Python code for implementing Linear regression and show its performance.**

CODE:

```
# Generate synthetic data
np.random.seed(42)

# Features: House size (in square feet)
X = 2.5 * np.random.randn(100) + 25

# Target: Price (in $1000s)
y = 100 + 5 * X + np.random.randn(100) * 10

# Convert the data into a pandas DataFrame
df = pd.DataFrame({'House_Size': X, 'Price': y})

# Print the first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X.reshape(-1, 1), y, test_size=0.2, random_state=42)

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict the house prices on the test set
y_pred = model.predict(X_test)
```

OUTPUT:

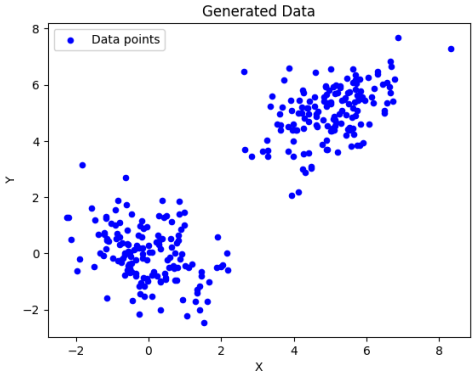


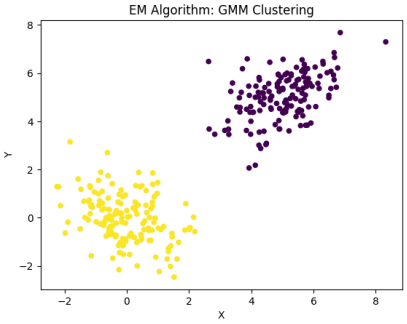
8. **Develop a Python code for implementing the EM algorithm with an example.**

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

# Generate synthetic 2D data (two Gaussian distributions)
np.random.seed(42)
mean1 = [5, 5]
cov1 = [[1, 0.5], [0.5, 1]]
data1 = np.random.multivariate_normal(mean1, cov1, 150)

mean2 = [0, 0]
cov2 = [[1, -0.3], [-0.3, 1]]
data2 = np.random.multivariate_normal(mean2, cov2, 150)

# Combine the two datasets
data = np.vstack((data1, data2))

# Plot the original data
plt.scatter(data[:, 0], data[:, 1], s=20, color='blue', label='Data points')
plt.title("Generated Data")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.show()
```

OUTPUT:





DAY-4 LAB EXPERIMENTS

1. **Can the breast cancer classification problem be solved using Naive Bayes classification print the 1st five rows. (b) Basic statistical computations on the data set or distribution of data (c) The columns and their data types**
   Detects null values in the dataset. If there is any null values replaced it with mode value (e) Split the data in to test and train
   evaluate the performance of the model by evaluation metrics such as confusion matrix.

   CODE:

```python
# Sample synthetic dataset
data = {
    'radius_mean': [17.99, 20.57, 19.69, 11.42, 20.29],
    'texture_mean': [10.38, 17.77, 21.25, 20.38, 14.34],
    'perimeter_mean': [122.8, 132.9, 130.0, 77.58, 135.1],
    'area_mean': [1001.0, 1326.0, 1203.0, 386.1, 1297.0],
    'smoothness_mean': [0.1184, 0.08474, 0.1096, 0.1425, 0.1003],
    'target': [1, 0, 1, 0, 1]  # 1 = Malignant, 0 = Benign
}

# Convert to DataFrame
df = pd.DataFrame(data)

# (a) Print the first five rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())

# (b) Basic statistical computations on the dataset
print("\nBasic statistics of the dataset:")
print(df.describe())

# (c) The columns and their data types
print("\nColumns and their data types:")
print(df.dtypes)
```

OUTPUT:

```
First 5 rows of the dataset:
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   target
0       1
1       0
2       1
3       0
4       1

Basic statistics of the dataset:
       radius_mean  texture_mean  perimeter_mean    area_mean  \
count     5.000000      5.000000        5.000000     5.000000
mean     17.992000     16.824000      119.676000  1042.620000
std       3.807929      4.495134       23.985368   388.421771
min      11.420000     10.380000       77.580000   386.100000
25%      17.990000     14.340000      122.800000  1001.000000
50%      19.690000     17.770000      130.000000  1203.000000
75%      20.290000     20.380000      132.900000  1297.000000
max      20.570000     21.250000      135.100000  1326.000000
```

2. **Implement a Python program for the most specific hypothesis using Find-S algorithm for the following given dataset and show the output:**

Size	Color	Shape	Class
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

CODE:

```
data = {
    'Size': ['Big', 'Small', 'Small', 'Big', 'Small'],
    'Color': ['Red', 'Red', 'Red', 'Blue', 'Blue'],
    'Shape': ['Circle', 'Triangle', 'Circle', 'Circle', 'Circle'],
    'Class': ['No', 'No', 'Yes', 'No', 'Yes']
}

# Create a DataFrame
df = pd.DataFrame(data)

# Initialize the most specific hypothesis
specific_hypothesis = ['0'] * (df.shape[1] - 1)  # Ignoring the Class column

# Iterate through the dataset to find the most specific hypothesis
for index, row in df.iterrows():
    if row['Class'] == 'Yes':
        for i in range(len(specific_hypothesis)):
            if specific_hypothesis[i] == '0':  # If it hasn't been specified yet
                specific_hypothesis[i] = row[i]  # Set to the current example
            elif specific_hypothesis[i] != row[i]:
                specific_hypothesis[i] = '?'  # Make it a wildcard

# Convert the result to a more readable format
result_hypothesis = [hypothesis if hypothesis != '0' else '?' for hypothesis in specific_hypothesis]
```

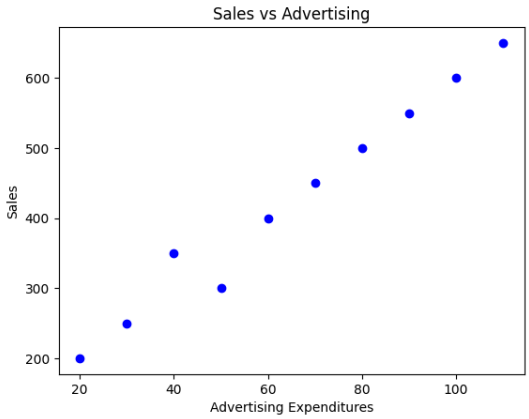
OUTPUT:

```
Most Specific Hypothesis:
['Small', '?', 'Circle']
```

3. **Develop a Python code for implementing Polynomial regression and show its performance (REPEATED)**
4. **Develop a Python code for implementing the KNN algorithm with an example.(REPEATED)**
5. **You are a data scientist at a retail company and your manager has asked you to create a model to predict future sales. The company has been collecting data on sales, and advertising expenditures, for the past 5 years. Your manager wants to use this information to forecast sales for the next quarter and make informed decisions about advertising and inventory.Your task is to build a predictive model that takes into account past sales data, and advertising expenditures, to forecast sales for the next quarter. You decide to use linear regression to build your model because it is a simple and interpretable method for predicting a continuous outcome.**
    a. **print the 1st five rows.**
    b. **Basic statistical computations on the data set or distribution of data**
    c. **the columns and their data types**
    d. **Explore the data using scatterplot**
    e. **Detects null values in the dataset. If there is any null values replaced it with mode value**
    f. **Split the data in to test and train**
    g. **Predict the model**

```
# Generate synthetic sales and advertising data
data = {
    'Sales': [200, 250, 300, 350, 400, 450, 500, 550, 600, 650],
    'Advertising': [20, 30, 50, 40, 60, 70, 80, 90, 100, 110]
}

# Create a DataFrame
df = pd.DataFrame(data)

# (a) Print the first five rows
print("First 5 rows of the dataset:")
print(df.head())

# (b) Basic statistical computations on the dataset
print("\nBasic statistics of the dataset:")
print(df.describe())

# (c) The columns and their data types
print("\nColumns and their data types:")
print(df.dtypes)

# (d) Explore the data using scatterplot
plt.scatter(df['Advertising'], df['Sales'], color='blue')
plt.title("Sales vs Advertising")
plt.xlabel("Advertising Expenditures")
```

OUTPUT:



6. **For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.**

Size	Color	Shape	Class
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

CODE:

```
# Define the dataset directly in the code
data = {
    'Size': ['Big', 'Small', 'Small', 'Big', 'Small'],
    'Color': ['Red', 'Red', 'Red', 'Blue', 'Blue'],
    'Shape': ['Circle', 'Triangle', 'Circle', 'Circle', 'Circle'],
    'Class': ['No', 'No', 'Yes', 'No', 'Yes']
}

# Create a DataFrame
df = pd.DataFrame(data)

# Separate positive and negative examples
positive_examples = df[df['Class'] == 'Yes']
negative_examples = df[df['Class'] == 'No']

# Initialize the most specific and general hypotheses
specific_hypothesis = ['0'] * (df.shape[1] - 1)  # Ignore the Class column
general_hypothesis = ['?'] * (df.shape[1] - 1)

# Find the most specific hypothesis
for index, row in positive_examples.iterrows():
    for i in range(len(specific_hypothesis)):
        if specific_hypothesis[i] == '0':
            specific_hypothesis[i] = row[i]
        elif specific_hypothesis[i] != row[i]:
```

OUTPUT:

```
Most Specific Hypothesis:
['Small', '?', 'Circle']

Most General Hypothesis:
['Big', '?', 'Circle']
```

7. **Develop a Python code for implementing Logistic regression and show its performance (REPEATED)**
8. **Develop a Python code for implementing the Naive Bayes algorithm with an example.(REPEATED)**

DAY-5 LAB EXPERIMENTS

1. **Julia is Botanist who is studying the Iris genus, She has collected the data of different the sepal length, sepal width, petal length, and petal width of various Iris flowers and wants to classify the flowers into their respective species based on their physical characteristics.Julia decides to compare the performance of different machine learning algorithms for this task. She splits her data into a training set and a test set and trains several models, including Decision tree classifier, Logistic Regression, KNN classifier. Julia wants to the performance measures based on accuracy and speed of execution. Help her do the comparison of the classification algorithms.**

CODE:

```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Logistic Regression": LogisticRegression(max_iter=200),
    "KNN": KNeighborsClassifier()
}

# Dictionary to hold results
results = {}

# Train and evaluate each model
for name, model in models.items():
    start_time = time.time()  # Start time for execution
    model.fit(X_train, y_train)  # Train the model
    y_pred = model.predict(X_test)  # Make predictions
    accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
    end_time = time.time()  # End time for execution
```

OUTPUT:

```
Model Performance Comparison:
Model                Accuracy   Execution Time (s)
Decision Tree        1.0000     0.0034
Logistic Regression  1.0000     0.0366
KNN                  1.0000     0.0075
```

2. **For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.**

Example	Citations	Size	In Library	Price	Editions	Buy
1	Some	Small	No	Affordable	Few	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

CODE:

```
class CandidateElimination:
    def __init__(self, data):
        self.data = data
        self.attributes = list(data[0].keys())
        self.attributes.remove('Buy')
        self.S = self.most_specific_hypothesis()
        self.G = self.most_general_hypothesis()

    def most_specific_hypothesis(self):
        return {attr: None for attr in self.attributes}

    def most_general_hypothesis(self):
        return {attr: '?' for attr in self.attributes}

    def train(self):
        for example in self.data:
            if example['Buy'] == 'No':
                self.S = self.remove_irrelevant_attributes(self.S, example)
            else:
                self.G = self.add_relevant_attributes(self.G, example)
        return self.S, self.G

    def remove_irrelevant_attributes(self, S, example):
        for attr in self.attributes:
            if S[attr] is None or S[attr] == example[attr]:
```

OUTPUT:

```
Specific Boundary (S): {'Citations': None, 'Size': None, 'In Library': None, 'Price': None, 'Editions': None}
General Boundary (G): {'Citations': '?', 'Size': '?', 'In Library': '?', 'Price': '?', 'Editions': '?'}
```

3. **Develop a Python code for implementing Polynomial regression and show its performance. (REPEATED)**
4. **Develop a Python code for implementing the KNN algorithm with an example. (REPEATED)**
5. **How is the Perceptron algorithm applied to the Iris flower classification problem?Rani is a botanist who is studying the Iris genus. She has collected data on the sepal length, sepal width, petal length, and petal width of various Iris flowers and wants to classify the flowers into their respective species based on their physical characteristics. Anna decides to use the Perceptron algorithm for this task.**

CODE:

```python
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, confusion_matrix

# Load Iris dataset
iris = load_iris()

# Define features (X) and target variable (y)
X = iris.data
y = iris.target

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize Perceptron classifier
clf = Perceptron(random_state=42, max_iter=1000, tol=1e-3)

# Train Perceptron model
clf.fit(X_train, y_train)

# Make predictions on test set
y_pred = clf.predict(X_test)
```

OUTPUT:

```
Accuracy: 0.6333333333333333
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0 11  0]]
```

6. **Implement a Python program for the most specific hypothesis using Find-S algorithm for the following given dataset and show the output:**

Example	Citations	Size	In Library	Price	Editions	Buy
1	Some	Small	No	Affordable	Few	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

CODE:

```python
def find_s(data):
    # Initialize most specific hypothesis with first positive example
    hypothesis = {}
    for attr, value in data[1].items():
        if attr != 'Buy':
            hypothesis[attr] = value

    # Iterate through remaining positive examples
    for example in data[2:]:
        if example['Buy'] == 'Yes':
            for attr, value in hypothesis.items():
                if example[attr] != value:
                    hypothesis[attr] = '?'  # Replace with '?' if values don't match

    return hypothesis


# Given dataset
data = [
    {'Citations': 'Some', 'Size': 'Small', 'In Library': 'No', 'Price': 'Affordable', 'Editions': 'Few', 'Buy': 'No'},
    {'Citations': 'Many', 'Size': 'Big', 'In Library': 'No', 'Price': 'Expensive', 'Editions': 'Many', 'Buy': 'Yes'},
    {'Citations': 'Many', 'Size': 'Medium', 'In Library': 'No', 'Price': 'Expensive', 'Editions': 'Few', 'Buy': 'Yes'}
]
```

OUTPUT:

```
Most Specific Hypothesis:
Citations: Many
Size: ?
In Library: No
Price: Expensive
Editions: ?
```

8. Develop a Python code for implementing to compare Linear and Logistic regression and show its performance (REPEATED)

9.Develop a Python code for implementing the EM algorithm with an example. (REPEATED)